Q1

class Solution:

def threeSumClosest(self, nums: List[int], target: int) -> int:

nums.sort()

n = len(nums)

diff = float('inf')

res = 0

for i in range(n - 2):

left = i + 1

right = n - 1

while left < right:

s = nums[i] + nums[left] + nums[right]

if abs(target - s) < diff:

diff = abs(target - s)

res = s

if s < target:

left += 1

else:

right -= 1

return res

Q2

class Solution:

def fourSum(self, nums: List[int], target: int) -> List[List[int]]:

nums.sort()

n = len(nums)

result = []

for i in range(n - 3):

if i > 0 and nums[i] == nums[i - 1]:

continue

for j in range(i + 1, n - 2):

if j > i + 1 and nums[j] == nums[j - 1]:

continue

l, r = j + 1, n - 1

while l < r:

s = nums[i] + nums[j] + nums[l] + nums[r]

if s == target:

result.append([nums[i], nums[j], nums[l], nums[r]])

while l < r and nums[l] == nums[l + 1]:

l += 1

while l < r and nums[r] == nums[r - 1]:

r -= 1

l += 1

r -= 1

elif s < target:

l += 1

else:

r -= 1

return result

Q3

class Solution:

def nextPermutation(self, nums: List[int]) -> None:

i=len(nums)-1

while i>0 and nums[i-1]>=nums[i]:

i=i-1

if i<1:

nums[:]=nums[::-1]

else:

j=len(nums)-1 #last element

#we have to find element from last that is greater than nums[i-1]

while nums[j]<=nums[i-1]:

j=j-1

nums[i-1], nums[j]= nums[j], nums[i-1]

nums[i:]=reversed(nums[i:])

Q4

class Solution:

    def searchInsert(self, nums: List[int], target: int) -> int:

        l = 0

        r =len(nums) - 1

        while l <= r:

            mid = (l+r) // 2

            if nums[mid] < target:

                l = mid +1

            elif nums[mid] > target:

                r = mid -1

            else:

                return mid

        return l

Q5

class Solution:

    def plusOne(self, digits: List[int]) -> List[int]:

        i = len(digits) -1

        while i >= 0:

            if(digits[i] == 9):

                digits[i] = 0

            else:

                digits[i]+=1

                return digits

            i-=1

        return[1] + digits

Q6

class Solution:

    def singleNumber(self, nums: List[int]) -> int:

        res = 0;

        for n in nums:

            res = n^res

        return res

Q7

class Solution:

def summaryRanges(self, nums: List[int]) -> List[str]:

if not nums: return nums

else:

a=b=nums[0]

out=[]

for i in nums[1:]:

if b == i-1:

b = i

else:

if a!=b: out.append(str(a)+"->"+str(b))

else: out.append(str(a))

a=b=i

if a!=b: out.append(str(a)+"->"+str(b))

else: out.append(str(a))

return out

Q8

def canAttendMeetings(intervals):

intervals.sort(key=lambda a: a.start)

for i in range(len(intervals)-1):

if intervals[i].end > intervals[i+1].start:

return False

return True